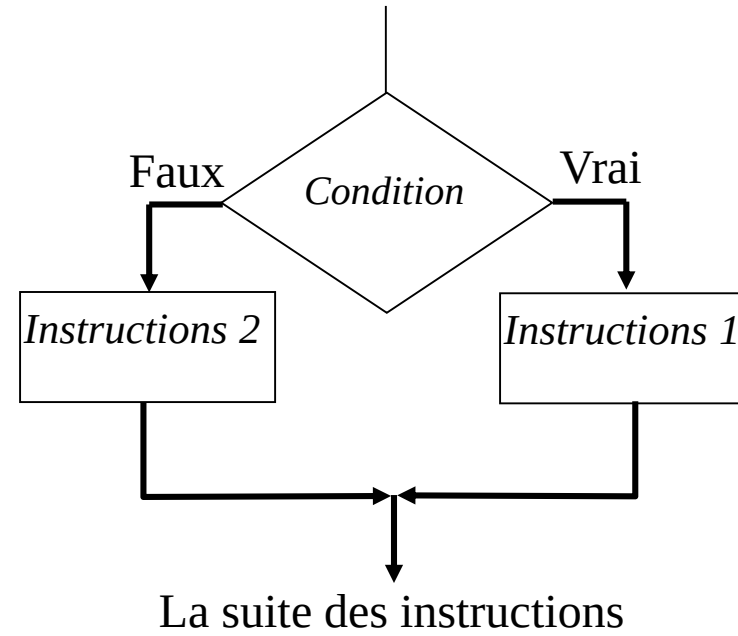


Algorithmes

Les conditions et les boucles



Les structures conditionnelles

SI, Sinon et Selon

Algorithme

c) Structure conditionnelle :

- L'exécution d'un bloc d'instructions peut être conditionnée par la vérification d'une condition.
- Une condition est une expression logique qui ne peut prendre que deux valeurs VRAI ou FAUX.
- Avant de voir comment on écrit une condition, il faut connaître les symboles de comparaison. Ces symboles sont indispensables pour réaliser des conditions.

$A=B$	A est égal à B
$A > B$	A est supérieur à B
$A < B$	A est inférieur à B
$A \geq B$	A est supérieur ou égal à B
$A \leq B$	A est inférieur ou égal à B
$A \neq B$	A est différent de B

Algorithme

c.1) La structure simple :

```
SI (condition) Alors  
    < instructions >  
Fsi;
```

- Si la condition (*condition*) est vérifiée, on exécute les instructions < *instructions* >, et on passe à la suite de l'algorithme. Si la condition n'est pas vérifiée, alors on passe à la suite de l'algorithme.
- **Exemple** : Ecrire l'algorithme qui affiche la valeur absolue d'un nombre entier A.

Algorithme

Algorithme ValAbs;

Var

A : entier

Début

Lire(A);

SI (A < 0) **ALORS**

A $\leftarrow$ A*(-1);

FSI

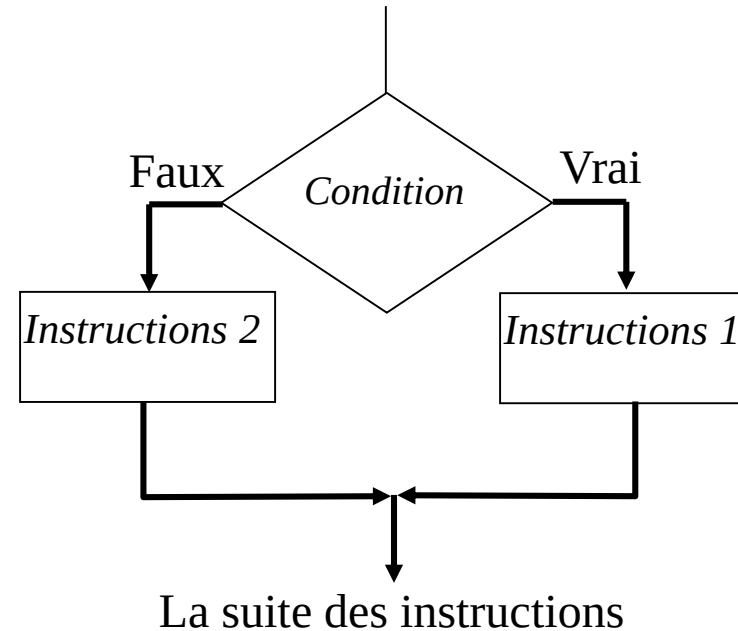
Ecrire ('la valeur absolue est :', A);

Fin.

Algorithme

c.2) La structure complete :

SI (*condition*) **Alors**
< *instructions 1* >
Sinon
< *instructions 2* >
Fsi;



- **Si** la condition (*condition*) est vérifiée, on exécute les instructions < *instructions 1* > et on passe à la suite de l'algorithme; **sinon** on exécute les instructions < *instructions 2* > et on passe à la suite de l'algorithme.

Algorithme

Exemple : Ecrire l'algorithme qui affiche le maximum de deux nombres entiers A et B.

Algorithme Maximum;

Var A, B, Max : entier;

Début

 Lire (A, B);

SI (A >= B) **Alors**

 Max ← A

SINON

 Max ← B;

FSI

 Ecrire ('le maximum est :', Max);

Fin

Algorithme

c.3) La structure imbriquée :

```
Si <Condition 1> Alors
    <Instructions1>
Sinon
    Si <Condition 2> Alors
        <Instructions2>
    Sinon
        ....
        ....
        ....
    Fsi ;
Fsi;
```

- Plusieurs conditions peuvent être imbriquées, il s'agit d'un choix multiple.

Algorithme

Exemple : donner un algorithme qui fait la comparaison de deux nombres entiers positifs $N1$ et $N2$.

Algorithme Comparaison;

Var

Début

Fin

Algorithme

Exemple : donner un algorithme qui fait la comparaison de deux nombres entiers positifs N1 et N2.

Algorithme Comparaison;

Var N1, N2 : entier;

Début

 Lire (N1, N2) ;

Fin

Algorithme

Exemple : donner un algorithme qui fait la comparaison de deux nombres entiers positifs N1 et N2.

Algorithme Comparaison;

Var N1, N2 : entier;

Début

 Lire (N1, N2) ;

Si (N1= N2) **Alors**

 Ecrire (N1,'=',N2)

Sinon

Si N1>N2 **Alors**

 Ecrire (N1,'>',N2)

Sinon

 Ecrire (N1,'<',N2);

Fsi ;

Fsi ;

Fin

Algorithme

c.4) La structure à choix multiple :

- Toutefois, la structure imbriquée peut nécessiter plusieurs répétitions. Pour éviter ces répétitions dans le test de la valeur d'une seule et même variable, on peut utiliser une autre structure. Cette structure particulière s'appelle "à choix multiple".

Selon <variable de choix>

Choix 1 : Séquence d'instructions 1;

Choix 2 : Séquence d'instructions 2;

.....

.....

Choix n : Séquence d'instructions n;

Fselon;

Algorithme

Exemple : affichage du mois de l'année.

Selon Mois

1 : Ecrire ('Janvier') ;

2 : Ecrire ('Février') ;

.....

12 : Ecrire ('Décembre')

Fselon

Les structures itératives

Les boucles

Algorithme

d) Les structures itératives (les boucles) :

- Une boucle est une technique permettant de répéter les mêmes instructions un certain nombre de fois.
- Nous allons voir 3 types de boucles :
 - Pour
 - Tant que
 - Répéter ... jusqu'à

Algorithme

d.1) la boucle « pour » :

- La boucle pour est très utilisée en programmation.
- **Syntaxe:**

```
Pour Compteur ← val_Initiale à val_Finale inc pas=Valeur_de_pas  
    Faire  
        <Bloc d'instructions>  
FPour
```

- **Compteur** est une variable de type entier (ou caractère). Il commence par **val_Initiale** et termine par **val_Finale** .

Algorithme

```
Lire (nb);  
Pour i:= 1 à nb-1  
    Faire  
        ecrire(i);  
Fpour;
```

- **Pas** est un entier qui peut être positif ou négatif. **Pas** peut ne pas être mentionné, car par défaut sa valeur est égal à 1. Dans ce cas, le nombre d'itérations est égal à

$$Val_Finale - Val_Initiale + 1.$$

- **Val_Initiale** et **Val_Finale** peuvent être des valeurs, des variables définies avant le début de la boucle ou des expressions de même type que compteur.

Algorithme

- **Remarque :**

Si la $\text{val_Initiale} < \text{val_Finale}$

Alors c'est une **Incrémentation** du compteur

Sinon c'est une **Décrémentation** du compteur

- **Exemple :**

Calcul de x à la puissance n (x^n) où x est un réel non nul et n un entier positif ou nul.

Algorithme

Algorithme puissance;

Var

Debut

Fin

Algorithme

Algorithme puissance;

Var x, puiss : réel;

 n, i : entier;

Debut

Fin

Algorithme

Algorithme puissance;

Var x, puiss : réel;
n, i : entier;

Debut

Ecrire (" Entrez la valeur de x ") ;
Lire (x) ;
Ecrire (" Entrez la valeur de n ") ;
Lire (n) ;

Pour i $\leftarrow$ 1 à n **faire**

puiss $\leftarrow$ puiss*x;

FinPour;

Ecrire (x, " à la puissance ", n, " est égal à ", puiss);

Fin

Algorithme

Algorithme puissance;

Var x, puiss : réel;
n, i : entier;

Debut

Ecrire (" Entrez la valeur de x ") ;

Lire (x) ;

Ecrire (" Entrez la valeur de n ") ;

Lire (n) ;

puiss $\leftarrow$ 1;

Pour i $\leftarrow$ 1 à n **faire**

puiss $\leftarrow$ puiss*x;

FinPour;

Ecrire (puiss);

Fin

Algorithme

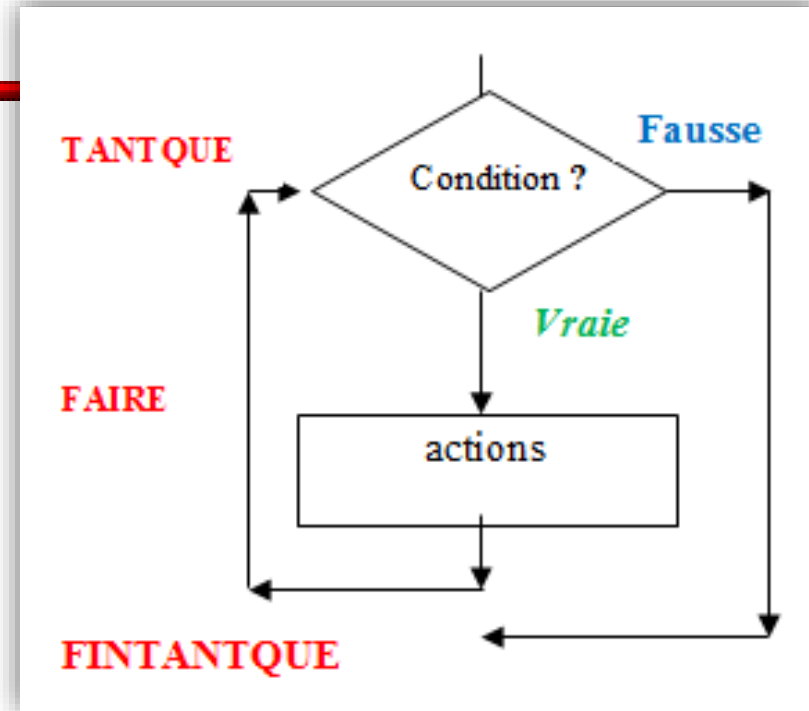
d.2) la boucle Tantque :

▪ Syntaxe :

Tant que (Condition) **faire**

< Instructions >

Ftantque;



- Selon cette boucle on veut dire, Tant que la condition est vraie, on répète l'exécution de bloc < instructions > et dès que la <Condition> devient fausse, on sort de la boucle. Si la condition est initialement fausse, le bloc d'instructions ne sera jamais exécuté.

Algorithme

Exemple : Ecrire un algorithme permettant de calculer la somme des N premiers nombres entiers positifs selon la formule $S=1+2+....+N$.

Algorithme somme;

Var

Debut

Fin

Algorithme

Exemple : Ecrire un algorithme permettant de calculer la somme des N premiers nombres entiers positifs selon la formule $S=1+2+....+N$.

Algorithme somme;

Var I,N,S : entier ;

Debut

Fin

Algorithme

Exemple : Ecrire un algorithme permettant de calculer la somme des N premiers nombres entiers positifs selon la formule $S=1+2+....+N$.

Algorithme somme;

Var I,N,S : entier ;

Debut

 Lire (N) ;

$S \leftarrow 0$;

$I \leftarrow 1$;

Fin

Algorithme

Exemple : Ecrire un algorithme permettant de calculer la somme des N premiers nombres entiers positifs selon la formule $S=1+2+....+N$.

Algorithme somme;

Var I,N,S : entier ;

Debut

 Lire (N) ;

$S \leftarrow 0$;

$I \leftarrow 1$;

Tantque (I <= N) **Faire**

$S \leftarrow S+I$;

$I \leftarrow I+1$;

FTantque;

 Ecrire ('la somme est :',S) ;

Fin

Algorithme

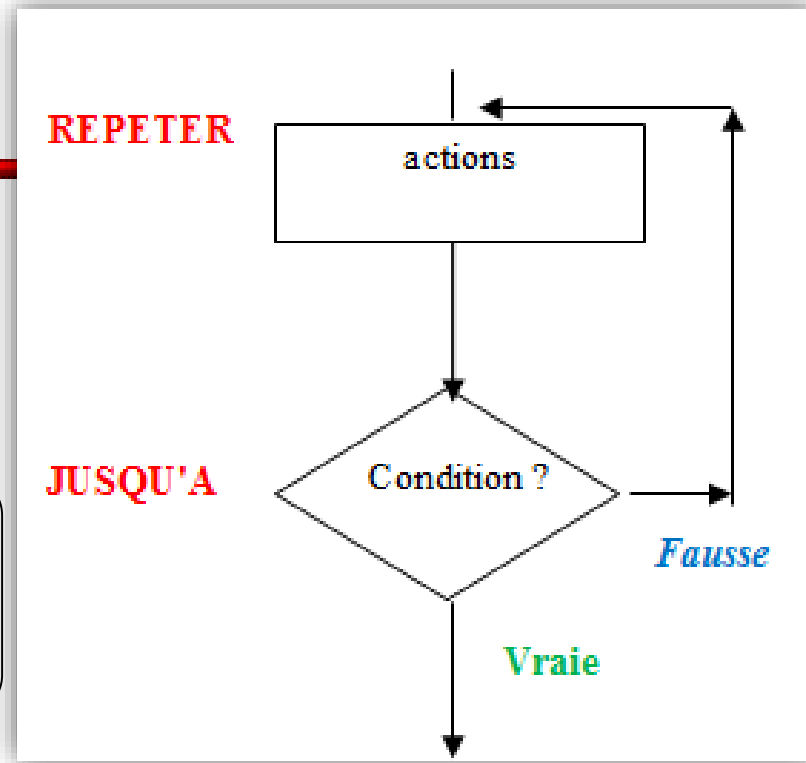
d.3) la boucle Répéter :

▪ Syntaxe :

Répéter

< Instructions >

jusqu'à (Condition)



- Ce type de boucle est très similaire à Tant que, bien qu'un peu moins utilisé en général.
- La seule chose qui change par rapport à Tant que, c'est la position de la condition. Au lieu d'être au début de la boucle, la condition est à la fin.

Algorithme

- la boucle ***Tant que*** pourrait très bien ne jamais être exécutée si la condition est fausse dès le départ. Par contre la boucle ***répéter*** s'exécutera au moins une fois.
- Il est donc parfois utile de faire des boucles de ce type, pour s'assurer que l'on rentre au moins une fois dans la boucle.
- **Exemple**

Répéter

Ecrire ('Entrer un nombre positif') ;

Lire(nombre) ;

Jusqu'à (nombre > 0)

Des questions?

